# Evaluation of Snort using rules for DARPA 1999 dataset

Ayushi Chahal[1], Dr. Ritu nagpal[2]
Department of Computer Science and Engineering,
Guru Jambheshwar University of Science & Technology,
Hisar, India.
[1]ayushichahal@gmail.com,[2]ritu_nagpal22@yahoo.co.in

*Abstract*— **Network security is main concern now-a-days and Snort is one of the advanced techniques that is used to tackle rising security threats over the internet. Snort is kind of Network Intrusion Detection System that allows user to write their own rules to detect different attacks over the network on the basis of their signatures and further gives freedom to users to handle these attacks in different ways. MIT-DARPA 1999 dataset (which consists of both normal and abnormal traffic) is used for evaluation of Snort in this paper. This evaluation is done with the help of the proposed detection rules. In this paper we have detected all kinds of attacks i.e. DOS, U2R, R2L, Probe and data attacks. These proposed rules results are further compared to the Detection Scoring Truth of DARPA 1999 dataset.**

***Index Terms*— *Network Intrusion Detection System (NIDS); Snort; detection rules; DARPA dataset*.**

## I. INTRODUCTION

Researches about network intrusion and intrusion detection began in the early 1980s by James Anderson. Intrusion detection system (IDS) in the recent years is generally considered to be the second line of defense after the firewall. IDS provide real-time protection ability and can intercept the invasion before the whole network system is endangered [1]. Different kinds of IDS are present in the market for detecting and protecting the data packet traffic over the network, like: Network based IDS (NIDS), Host based IDS (HIDS).

### *A. SNORT-IDS*

Snort is one of the famous and most effectively used NIDS against intruders.[2] It is signature based NIDS. Snort is an open source network intrusion prevention and detection utilizing a rule-driven language. Snort is available under the GNU (General Public License) [3].

Snort is a cross-platform operating system developed by Martin Roesch in 1998 [4]. He further found Sourcefire in 2001 [5] which created a commercial version of Snort having GPLv2+. With the acquisition of Sourcefire in October 2013, Snort is now one of the technologies used in Cisco products.

Snort++ is an updated version of the Snort IPS (intrusion prevention system).It uses friendly design with build in documentation and configuration. In it command shell allows interaction with running instance of snort. It provides facility of Auto-Detection of all protocols on all ports. It support multiple packet processing threads. It uses simplified rule language. It support sticky buffers in rules. It auto-detect services for portless configuration. It makes key components pluggable. [15]

### *B. DARPA dataset*

DARPA dataset is of interest to all researchers working on intrusion detection. It had been created by MIT Lincoln Laboratory IDS evaluation methodology. Such evolution was carried out in 1998 and 1999 which result out in form of: “1998 DARPA Intrusion Detection Evaluation Data Sets” and “1999 DARPA Intrusion Detection Evaluation Data Sets”, “2000 DARPA Intrusion Detection Scenario-Specific Datasets” of experiments run in 2000.

#### *1) DARPA dataset 1999*

First DARPA dataset 1998 came into existence, after some evolution in it 1999 DARPA Intrusion Detection became the dataset of interest to all the researchers. For evaluation of dataset DARPA 1999, it is divided into two parts [6]:

- Real-time Evaluation
- Off-line Evaluation

IDSs were tested as a part of real-time evaluation, off-line evaluation or both.Data collected under DARPA 1999 is of five weeks. Over all data inside this dataset is considered under two phases i.e. **Training dataset** and **Testing dataset**.

First three week data are Training data. In 1999, IDSs were trained with the help of dataset 1998 as well as dataset of 1999. Fourth week dataset and fifth week dataset are used as Testing data.

DARPA 1999 dataset contains very limited number of attacks that are detectable with the fixed signature. These attacks are divided into four categories. Detail of every category of attack is as follows:

1. DOS (Denial Of Service) attack: This type of attack occurs when legitimate users are not able to use computing and memory resources because intruder makes these resources too busy to handle authorized requests. Different kind of DOS attacks are shown below by table 1:

Table 1: Different kind of DOS attacks

| | | | |
|---|---|---|---|
| I. | Apache | II. | selfping |
| III. | appoison | IV. | Smurf |
| V. | Back | VI. | sshprocesstable |
| VII. | crashiis | VIII. | syslogd |
| IX. | dosnuke | X. | tcpreset |
| XI. | Land | XII. | Teardrop |
| XIII. | mailbomb | XIV. | udpstrom |
| XV. | SYN Flood | XVI. | warezmaster |
| XVII. | Ping Of Death | XVIII. | warezclient |
| XIX. | Process Table | | |

2. Remote to Local (R2L) attack: In it attacker who do not have account on the remote node sends packet to that node through the network. Attacker then exploits some vulnerability to gain access as a local user to that remote node. There are very difficult to detect as they involve both network level features such as “duration of connection” and “service requested” and host level features like “number of failed login attempts”. Different kind of R2L attacks are given below by table 2:

Table 2 : Different kind of R2L attacks

| | | | |
|---|---|---|---|
| I. | Dictionary | II. | ftp-write |
| III. | Guest | IV. | httptunnel |
| V. | imap | VI. | Named |
| VII. | ncftp | VIII. | netbus |
| IX. | netcat | X. | phf |
| XI. | ppmacro | XII. | sendmail |
| XIII. | sshtrojan | XIV. | xlock |

3. User to Local (U2R) attack : The attacker starts as a normal user on the system and becomes root user by gaining the root access through exploiting vulnerabilities. It involves exploitation on semantic details that’s why these attacks are difficult to capture at early stage. It features such as “number of file creations” and “number of shell prompts invoked”. Different kind of U2R attacks are shown by table 3 below:

Table 3 : Different kind of U2R attack

| | | | |
|---|---|---|---|
| I. | anypw | II. | casesen |
| III. | Eject | IV. | ffbconfig |
| V. | fdformat | VI. | loadmodule |
| VII. | ntfsdos | VIII. | Perl |
| IX. | Ps | X. | sechole |
| XI. | xterm | XII. | yaga |

4. Probe attack: The attacker scans the network of computers to collect information or to find known vulnerabilities. The attacker can use this information to exploit the nodes over this network. Hence, basic connection level features such as the “duration of connection” and “source bytes” are significant while features like “number of files creations” and number of files accessed” are not expected to provide information for detecting probes. Different kind of Probe attacks are shown by table 4 below:

Table 4 : Different kind of Probe attack

| | | | |
|---|---|---|---|
| I. | insidesniffer | II. | ipsweep |
| III. | is_domain | IV. | mscan |
| V. | ntinfoscan | VI. | nmap |
| VII. | quesco | VIII. | resetscan |
| IX. | Saint | X. | Satan |

### *2) DARPA dataset 2000*

After improvement in the DARPA 1999 dataset, DARPA 2000 has evolved. DARPA 2000 is a simulated network which is divided into three segments:

1 network outside Air Force Base (AFB)

2 network inside AFB

3 DMZ network which connects both inside and outside networks of AFB.

It includes two attack scenarios: LLDOS1.0 and LLDOS2.0.2. [6]

## *II. BACKGROUND AND RELATED WORK*

Martin Roesch gave the clear-cut difference between snort and tcpdump. He stated the basic working model of snort. According to him, snort consists of 3 primary subsystem: *the packet decoder, the detection engine and the logging & alerting system*. He gave a simple way to write snort rules. It has become a small, flexible and highly capable system that is used all around the world on both small and large scale network. He also define some applications of snort in his paper [4] like Snort can be used to characterize the signature of the attack, Snort can be used as Honey-pot monitors, Snort can help in “focused monitoring”.

Extended form of Snort architecture is presented by Kurundkar G.D *et al*.[7] as they gave snort component architecture with six components, namely : *Packet Decoder, Preprocessor, The Detection Engine, Logging and Alerting System, Output Modules*. They have explained different type of intrusion detection system like NIDS (Network Intrusion Detection and Prevention System), NBA(Network Behavior Analysis System), HIDS(Host Intrusion Detection System) and IDPS(Intrusion Detection and Prevention System). IDPS provide multiple detection methods: Signature based, Statistical Anomaly based, Stateful Protocol Analysis IDPS.

The best approach to any organization to perform penetration testing is to write snort rules to protect against attack. There is no alternative of secure coding. Alaa El- Din Riad *et al.*[8] presented a new frame work that is designed with using data visualization technique by using Jquery & php for analysis and visualizes snort result data for user. They presented a new way to represent Snort rule in form of rule header and rule option. Rule header contains information about what step should be taken if all the content in rule option matches.

Intrusion Detection rules such as Snort rules are increasingly becoming complicated and massive these days.[1] presented an innovative way which will largely enhance the detection efficiency in both space and time aspect. It gave an innovative way to organize rule in form of a three dimensional list. In it rule is firstly divided into various categories like alert, logs, or pass type and then further compartmentalized by different protocol types which is further divided into RTN (Rule Tree Node) i.e. header and their two pointers and OTN (Option Tree Node) i.e. body section and pointer to other body section, as that was divided in Two-dimensional implementation.

A new method for driving and testing malicious behavior of detection rule is introduced by Raimo Hilden *et.al.* which group the rules by their shared contents and extracts the most general rule of each group to optimized rule base. These pruned rules are stored for diagnostics. This method also maintains logs of which rule belong to which generalization. [9] uses rule generalization and rule base optimization. Key relation between signatures, rules and traffic packets must be maintained to avoid false positives and negatives in rule engine. It used content descriptor, connection descriptor, Match function to optimize the any rule R. It used signature parser, signature verifier, Signature Syntax Warning Logs, Rule Generator, Rule Verifier, Rule Syntax Warning Logs, Rule Purifier, Substitution Table in its architecture. For testing its results it used two methods: *dummy method*, *novel method*. In dummy method, pruning was done only on identical rules, and there was no proper linkage between rules and signature. In novel method, each file is given a signature with a unique identifier, in it they verified syntax of each signature. It greatly assists the experts in work as now they do not need to manually inspect signature, rules and traffic packet, they can now concentrate on automatically detected and reported issues.

Snort works on the signatures usually engineered based on experience and expert knowledge. It requires long development time. [10] gave approaches for an automated re-use of design of existing signatures. Sebastian Schmerl *et al.* showed their re-use approach for single-step signatures used by Snort. After selecting the related signatures with the help of signature generalization called abstraction, the engineer can look for similarities. Abstraction can be accomplished by iterative applying Transformations. This will result in Abstraction Tree. Each Transformation is weighted by a metric which defines similarity. The signature with lowest abstraction degrees are selected to understand nature of attack.

To improve Snort rules for Probe attacks N. Khamphakdee *et al.* uses MIT-DARPA 1999 data set. Firstly analysis of existing Snort-IDS rules was done to improve the proposed Snort-IDS rules. Secondly, WireShark software was applied to analyze data packets form of attack in data set. Finally, the Snort-IDS was improved, and it can detect the network probe attack. In [11] Probe attacks are divided into six types based on its nature. Comparison of Snort rules with the Detection Scoring Truth on the basis of efficacy of detection attacks is done and results of the tested Snort-IDS rules confirm that the proposed Snort-IDS can correctly detect 100% of the network probe attacks. Regarding to the comparative analysis with the notification Detection Scoring Truth, the detection number of the proposed Snort-IDS rules are more than the detection scoring truth. Because, some moments of the attacks had occurred in several times and attack occur in several time but the Detection Scoring Truth identify as one time.

[12] has shown that any rule that has one or more content matches in it has a fast pattern associated with it .The string that Snort puts into its fast pattern matching engine to begin the process of detection is chosen somewhat intelligently by Snort itself. This pattern is usually the longest string in a rule, because the longer the string is, the faster a rule will be. The goal of a rule-writer should be to choose a fast pattern, if one can generate an alert for most of the times when he/she enter a rule, then he/she has successfully targeted his/her detection, and written a rule with the minimum possible performance impact on Snort.

Most of the IDS researchers prefer to work on DARPA dataset with Snort. S. Terry *et al.* used tcpdump files as input to the Snort which was configured with all rules and alert file was produced for each tcpdump file. He analyzed performance of Snort with DARPA dataset 1999 on the basis of number of malicious connection detected. [13] Shows that for *DOS attack*, Snort performed best on back and land attack while it does not perform up to the mark for Smurf, Syslog and teardrop attack. In case of *R2L attack*, Snort performed best in case of phf attack with no false positive and performs worst in case of spy, warez, ftp, warezclient, warezmaster attack with low true positive rate. Snort did not performed that well for *U2R attack* as compared to other three attacks, yet it gives impressive results by reducing few false positives. In case of *probe attack*, threshold for detecting these attacks is very low in corresponding Snort rules and hence, majority of ipsweep connections was detected.

A. Saboor *et al*. evaluate Snort against DARPA attack DDoS attack with different hardware configurations. He used three test benches with different configurations but having Linux installed on each test bench as operation system.

- In these test benches, he used Hping for DDoS attack simulation tool, Hping and Ostinato for background traffic generation tool, Hping and Teamviwer for DoS verification tools.
- For attack, he used two scenarios i.e. attack scenario 1 having attack traffic only and attack scenario 2 having mix traffic.
- For evaluation matrices, he used maximum packet rate, resource availability, throughput, Snort rate filter option.

From his experiment [14] he concluded that in terms of detection capability of Snort no test bench outperformed other due to lack of Snort signature database and rate filtration. Using more RAM improved the performance of Snort. Hardware implementation decreased drop of packet to atmost 50%, but it did not show any improvement in packet handling and attack detection capability of Snort.

## III. *CREATING SNORT RULES*

### A. Experimental Setup

Rules are designed to detect different type of attacks with the help of their signatures. These rules are made with the help of Snort 2.9.8.3 with DAQ 2.0.6 version. It operated on Ubuntu 14.04 LTS operating system using Intel Pentium processor.

### B. SNORT installations

We install snort in Ubuntu in root directory with the help of installation manual of sublime roots. At the same time, we also install DAQ library in root.

First we install pre-requisites for snort like libpcap, libdumbnet, libpcre etc. Then we install and configure snort. After that, we create some directories like /etc/snort to configure and write rules, /var/log/snort which is used to store alerts generated during process. Also we adjusted paths of the local rules and white lists and black list rules.

### C. Dataset

DARPA 1999 dataset is used to make rules. As described above, DARPA 1999 have five weeks data. From which week 1, week 2, week 3 are training dataset. Since, we work on NIDS so we opted for week 2 dataset as a training dataset, because only this dataset contains labeled attacks. So with the help of week 2 dataset we made different rules. These rules are then tested on the testing dataset i.e. week 4 and week 5 dataset, to get our results. We have utilized *.tcpdump* file format by choosing *inside.tcpdump* and *outside.tcpdump* files in week 4 and week 5.

### D. Proposed SNORT rules

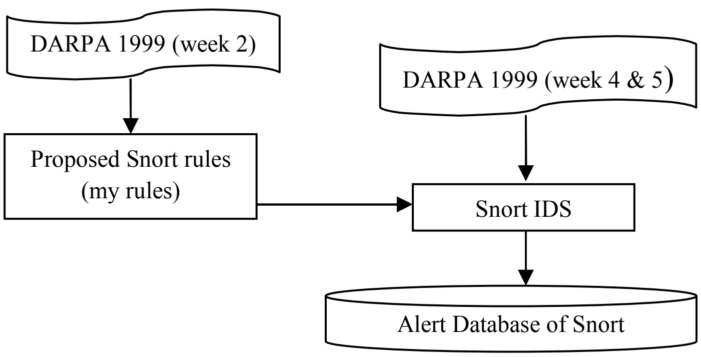


Fig 1 : Flow diagram shows making of SNORT rules

Week 2 dataset consists of some labeled attacks with their signatures inside, while week 1 and week 3 do not have any labeled attacks. So dataset like week 1 and week 3 can be used for IDS like anomaly detection. Since, snort is a NIDS and can detect attack on the basis of some content or signature of the attack inside the packet. With the help of these signatures of different attacks, different rules are made.

Hence we have used week 2 dataset as training data to create our rules. We took some signatures for each attack type and each attack type is given its corresponding *classtype* in which it fits best. Every rule is assigned a unique *sid* number.

We have generalized these rules by using *"any"* keyword in place of source and destination address places as well as for source and destination Port address places, so that they can send alert for all kind of Source and Destination addresses and port numbers. Some of the rules which are used to get final results are showen below:-

| |
|---|
| alert udp any any > any any (msg:"NTinfoscan attack which is kind of Probe is detected"; content:"NTinfoscan"; nocase; sid:10000001; rev:01; reference:url, https://www.ll.mit.edu; classtype:attempted<br>recon;) |
| alert udp any any > any any (msg:"pod attack which is a type of DOS is detected"; content:"pod"; nocase; sid:10000002; rev:01; reference:url, https://www.ll.mit.edu; classtype:denial-of-service;) |
| alert udp any any > any any (msg:"back attack which is type of DOS attack is detected"; content:"back"; nocase; sid:10000003; rev:01; reference:url,https://www.ll.mit.edu; classtype:denial-of-service;) |
| alert icmp any any > any any (msg:"ipsweep attack which is kind of Probe attack is detected"; content:"\|00 00 00 00 00 00 00 00 00\|"; nocase; sid:10000049; rev:01; reference:url, https://www.ll.mit.edu; classtype:icmp event;) |
| alert tcp any any > any any (msg:"NTinfoscan attack which is kind of Probe attack is detected"; content:"NTinfoscan"; nocase; sid:10000017; rev:01; reference:url, https://www.ll.mit.edu; classtype:tcp-connection;) |
| alert tcp any any > any any (msg:"pod attack which is a type of DOS attack is detected"; content:"pod"; nocase; sid:10000018; rev:01; reference:url, https://www.ll.mit.edu; classtype:denial-of-service;) |
| alert tcp any any > any any (msg:"back attack which is type of DOS is attack detected"; content:"back"; nocase; sid:10000019; rev:01; reference:url, https://www.ll.mit.edu; classtype:denial-of-service;) |
| alert tcp any any > any any (msg:"crashiis attack which is kind of DOS attack is detected"; content:"crashiis"; nocase; sid:10000051; rev:01; reference:url, https://www.ll.mit.edu; classtype:denial-of- service;) |
| alert ip any any > any any (msg:"NTinfoscan attack which is kind of Probe attack is detected"; content:"NTinfoscan"; nocase; sid:10000033; rev:01; reference:url, https://www.ll.mit.edu; classtype:tcp-connection;) |
| alert ip any any > any any (msg:"pod attack which is a type of DOS attack is detected"; content:"pod"; nocase; sid:10000034; rev:01; reference:url, https://www.ll.mit.edu; classtype:denial-of-service;) |

## *IV. PERFORMANCE EVALUATION*

This section describes the experimental evaluation of the SNORT rules which are created to compare the detection performance.

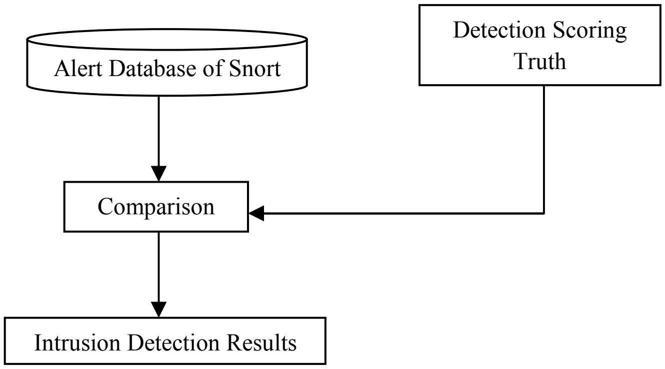


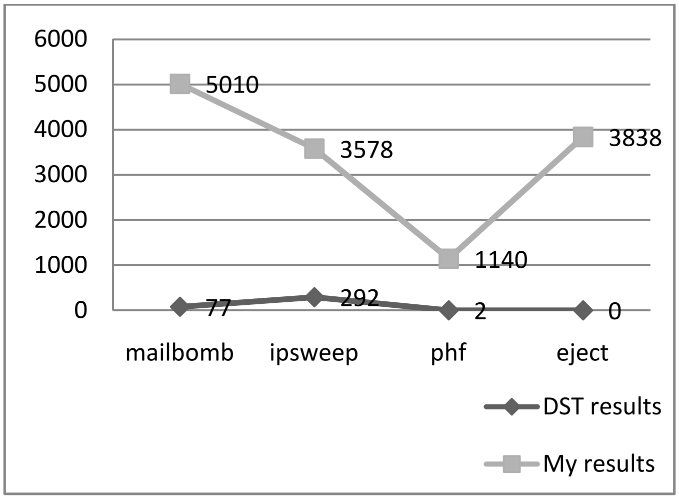
Fig 2: results of newly created rules compared with detection Scoring Truth

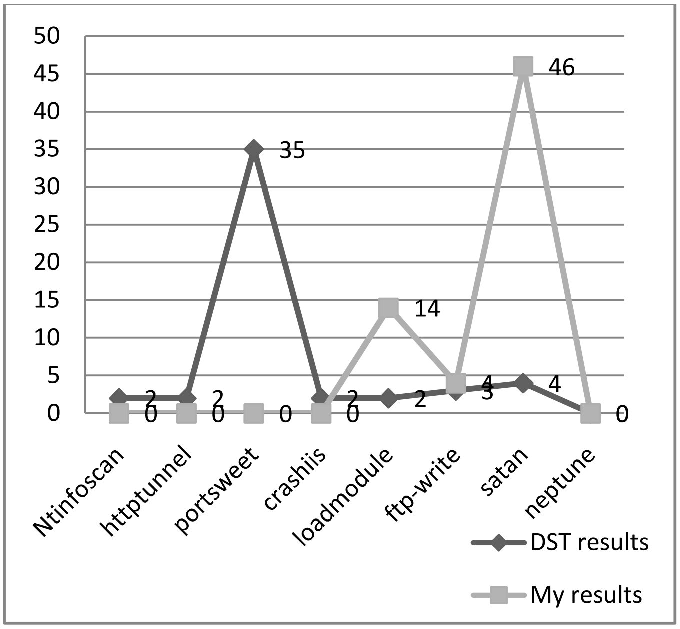
Figure 2 is a flow diagram shows the procedure of attack detection comparison of created rules with Detection Scoring Truth. Notifications that we get in alert database are compared with actual attack in Detection Scoring Truth.

We have used different type of attacks like Dos, U2R, R2L, Probe attacks to get our results and evaluate the performance. Here are the results shown in the form of graph. Graph is between the number of alerts we get through the SNORT rules (when applied on Week 4 and Week 5 data set of DARPA 1999) and Detection Scoring Truth.

1. Comparative results of week 4 with Detection Scoring Truth (DST) :

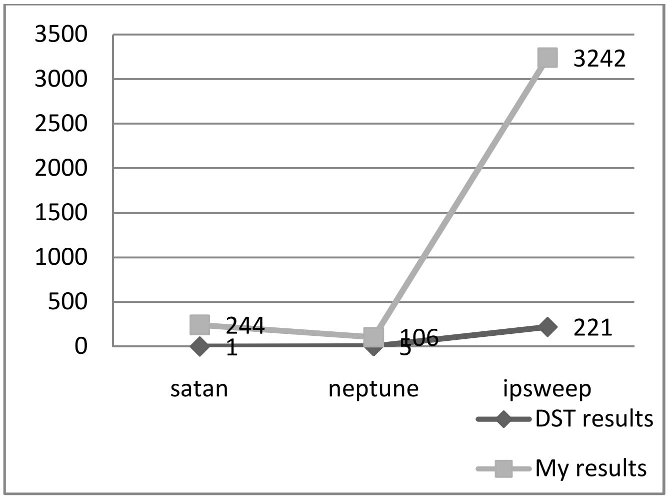
Here, in the graphs shown below result we can see that Data Scoring Truth of DARPA 1999 detects more attacks for NTinfoscan, httptunnel, portsweep, crashiis attack but for loadmodule, ftp-write, satan my results are far better than that of DST results.

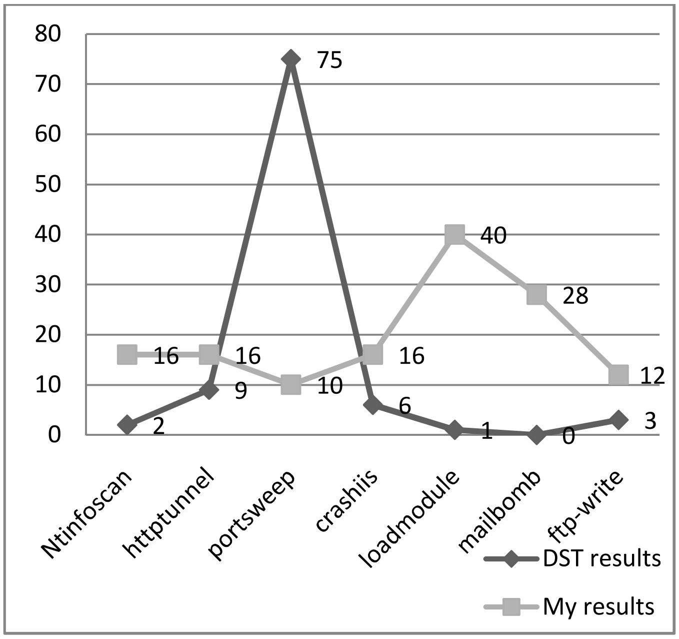




2. Comparative results of week 5 with Detection Scoring Truth (DST) :

Except portsweep attack all the remaining attack's rules in above graph namely NTinfoscan, httptunnel, crashiis, loadmodule, mailbomb, ftp-write performed well and give good results by giving appropriate amount alerts after detecting corresponding attacks.





## *V. CONCLUSION AND FUTURE DIRECTIONS*

We have created some Snort rules that are used to detect these signature based attacks. These rules also classify attacks according to their characteristics into different classtypes. DARPA dataset is considered as dataset of interest for intrusion detection researchers. So, we used DARPA training dataset to create Snort rules for different attacks. These rules are used in generalized form so that it can maximize the alert detection. This generalization is necessary so that rules can detect novel attacks. This generalization is achieved by relaxing some of the conditions on the rule. These rules are then tested on DARPA testing dataset which provide good results when compared to Detection Scoring Truth of DARPA dataset. Overall proposed rules performed very well as compared to DST rules.

There are several future directions for research:

*First*, there is need to improvise these rule with respect to false alerts. As we have discussed different factors for reducing false alerts like rule generalization, clustering method, alert correlation, feature frequencies, classification methods, data mining methods and neuro fuzzy method. We have only used rule generalization method for it. In future, we will be using any of these methods for reducing false alerts generated by Snort rules. *Second*, we have used static dataset "DARPA" for creating and testing our rules for different labeled attacks. In future, we will apply these rules on some dynamic dataset like BSNL server or some website data so that these rules can be made more efficient in all perspectives. *Third,* we will be using Snort++ or Snort 3.0 in future for further advancement in our rules to detect intrusions over the network. Snort++ alfa version is out now in the market for testing purpose.

## *REFERENCES*

[1] J. Kuang, L. Mei, and J. Bian, "An innovative implement in organizing complicated and massive intrusion detection rules of IDS," in *2nd International Conference on Cloud Computing and Intelligent Systems*, Hangzhou, Vol. 03, pp. 1328–1332, 2012.

[2] Ritu Makani, Yogesh Chaba "Analysis of Security Techniques for Computer Networks*",* in *International Journal of Engineering Research in Computer Science and Engineering*, Vol. 1, pp. 1-3, 2014.

[3] "SNORT Users Manual 2.9.8.2." [Online]. Available: http://manual-snort-org.s3-website-us-east-1.amazonaws.com/.

[4] M. Roesch and S. Telecommunications, "Snort - Lightweight Intrusion Detection for Networks", in *Proceedings of USENIX LISA,* Seattle, Washington, USA, Vol. 99, No. 1, pp. 229–238, 1999.

[5] E. Kostlan, (2015), "Intermediate - Snort Implementation in Cisco Products", *CiscoLive 365*.[online]. Available: http://www.ciscolive.com/online/connect/sessionDetail.ww?SESSION_ID=83682.

[6] "MIT Lincoln Laboratory: DARPA Intrusion Detection Evaluation." [Online]. Available: https://www.ll.mit.edu/ideval/data/2000data.html. [Accessed: 09 Apr-2016].

[7] G.D. Kurundkar, N.A. Naik and S.D. Khamitkar, "Network Intrusion Detection using SNORT", in *International Journal Of Engineering Research and Application*, Vol. 2, Issue 2, pp. 1288-1296, 2012.

[8] Riad, Alaa El-Din, Ibrahim Elhenawy, Ahmed Hassan, and Nancy Awadallah "Using Jquery with Snort to Visualize Intrusion.", in *International Journal of Computer Science*, Vol. 9, Issue 1, No. 3, pp. 486-491, 2012.

[9] R. Hilden and K. Hatonen, "A Method for Deriving and Testing Malicious Behavior Detection Rules," in *IEEE Trustcom/BigDataSE/ISPA*, Helsinki, Vol. 1, pp. 1337-1342, 2015.

[10] S. Schmerl, H. Koenig, U. Flegel, M. Meier, and R. Rietz, "Systematic Signature Engineering by Re-use of Snort Signatures," in *Annual Computer Security Applications Conference,* Anaheim, California, pp. 23–32, 2008.

[11] N. Khamphakdee, N. Benjamas, and S. Saiyod, "Improving Intrusion Detection System based on Snort rules for network probe attack detection," in *2nd International Conference on Information and Communication Technology*, pp. 69–74, 2014.

[12] A. Kirk, "Using Snort Fast Patterns Wisely For Rules", *Cisco Talos Blog* 2010.

[13] S. Terry and B. J. Chow, "An assessment of the DARPA IDS evaluation dataset using Snort", in *UCDAVIS department of Computer Science,* Vol. 1, pp. 22-41, 2007.

[14] A. Saboor, M. Akhlaq, and B. Aslam, "Experimental evaluation of Snort against DDoS attacks under different hardware configurations", in *2nd National Conference Information Assurance*, Rawalpindi, pp. 31–37, 2013.

[15] "Sourcefire VRT: Focused on protecting your network", *Sourcefire Whitepaper*, 2012

**Ayushi Chahal** received M.Tech degree in Computer Science and Engg. from Guru Jambheshwar University of Science and Technology, Hisar , Haryana , India in 2016. She has received her B.Tech degree in Computer Science and Engg. from Bhagat Phool Singh Mahila Vishwavidyalaya, Sonipat, Haryana, India. Her research interest includes Network Security, Big Data, IoT.

**Ritu Makani** is a Associate Professor in Computer Science and Engg. having 14 years of teaching and research experience. She has received her PhD. degree in 2014 from Guru Jambheshwar University of Science and Technology, Hisar, Haryana, India. She has done her M.Tech in 2002 from Guru Jambheshwar University of Science and Technology, Hisar, Haryana, India. Her core area of interest in research is Network Security.